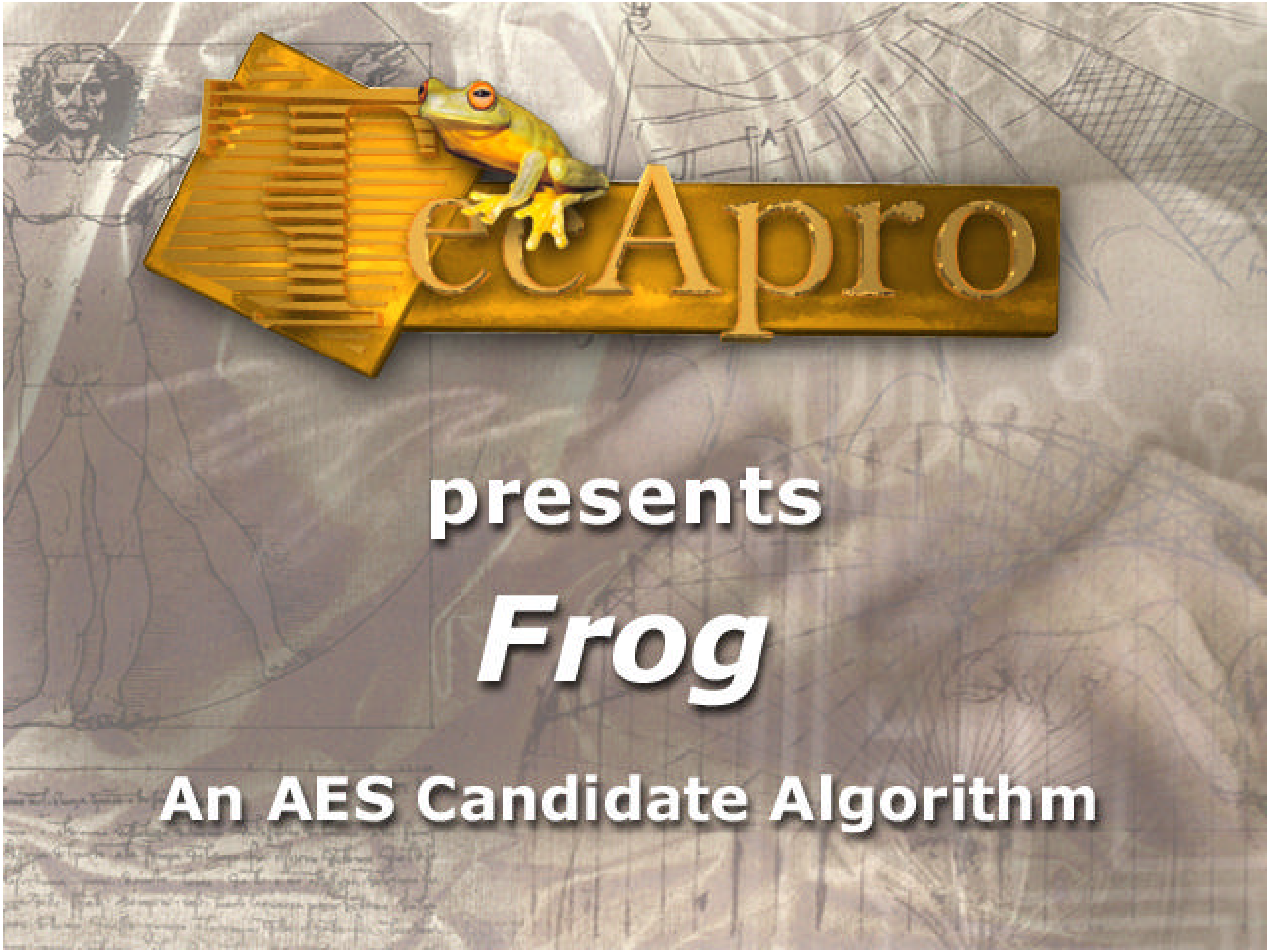
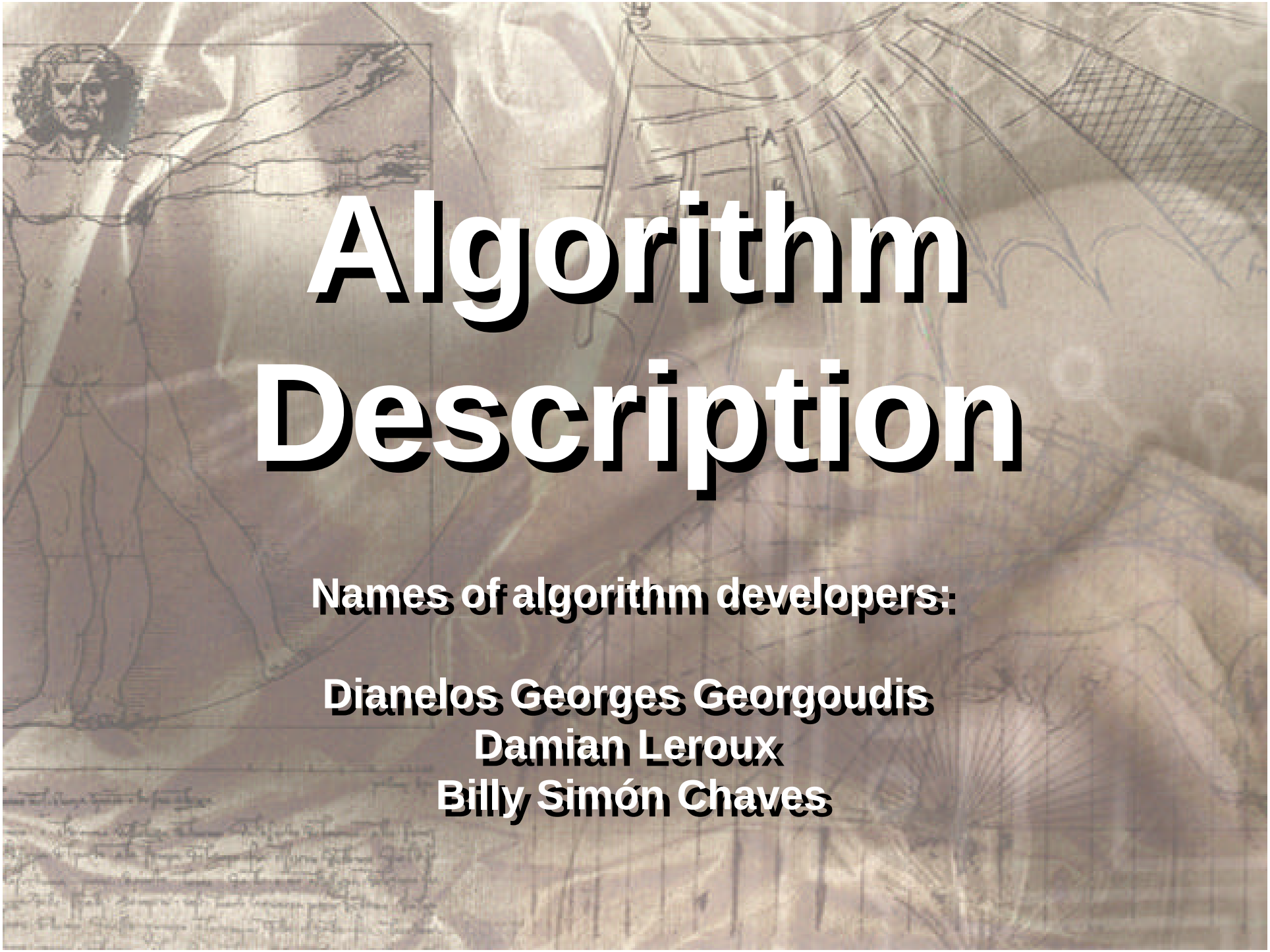

ecApro
presents
Frog
An AES Candidate Algorithm



Algorithm Description

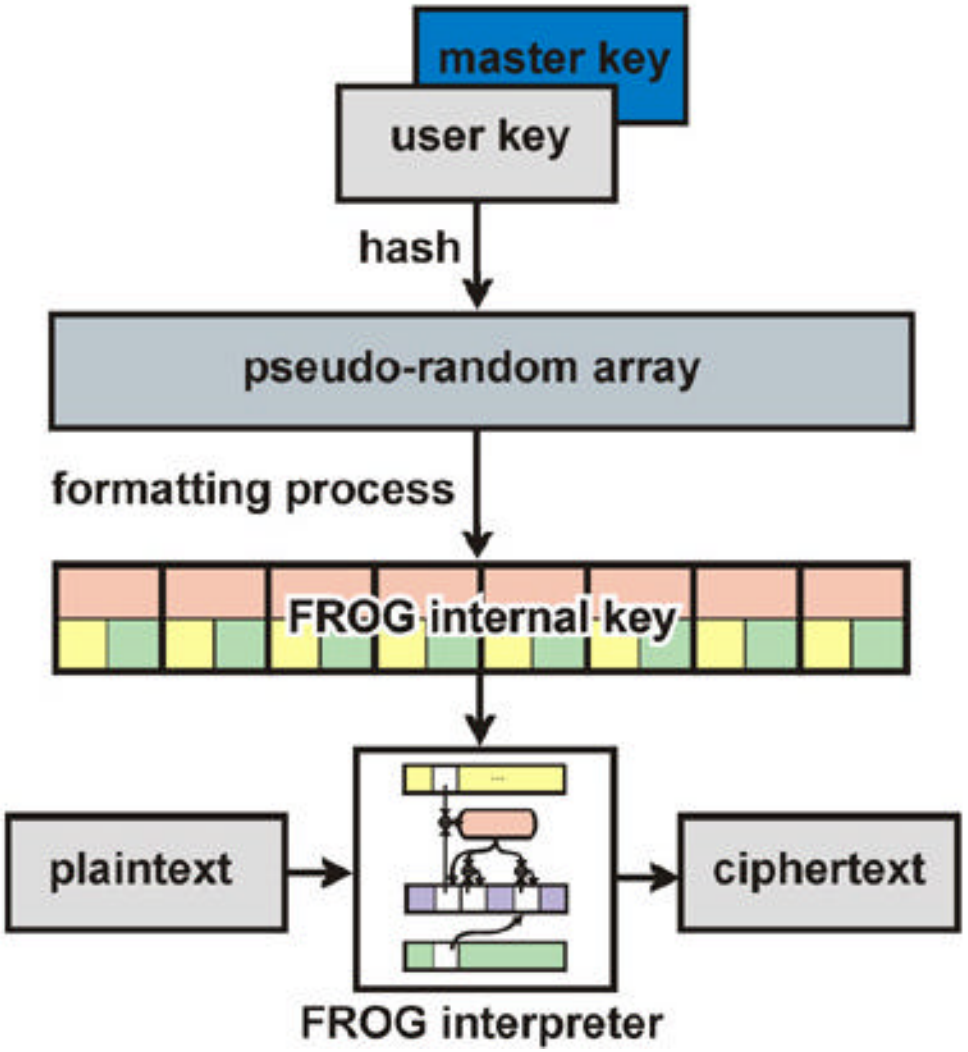
Names of algorithm developers:

Dianelos Georges Georgoudis
Damian Leroux
Billy Simón Chaves

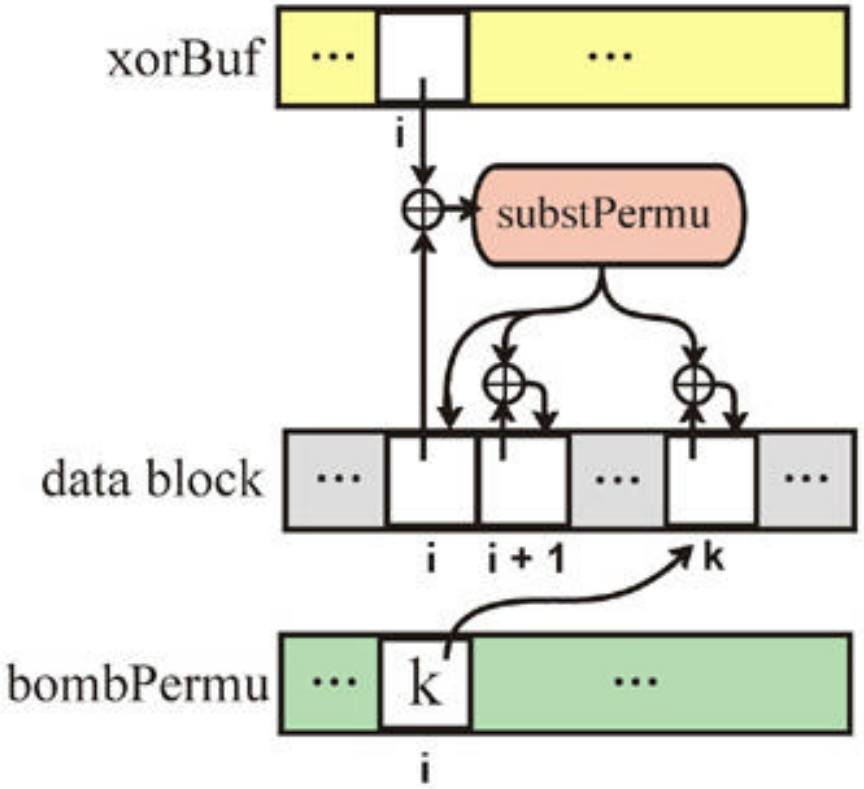
FROG encrypts a data block by interpreting a sequence of primitive invertible instructions coded into the internal key.
To decrypt a block, this process is simply run in reverse, thus effectively un-doing the encryption.

TecApro

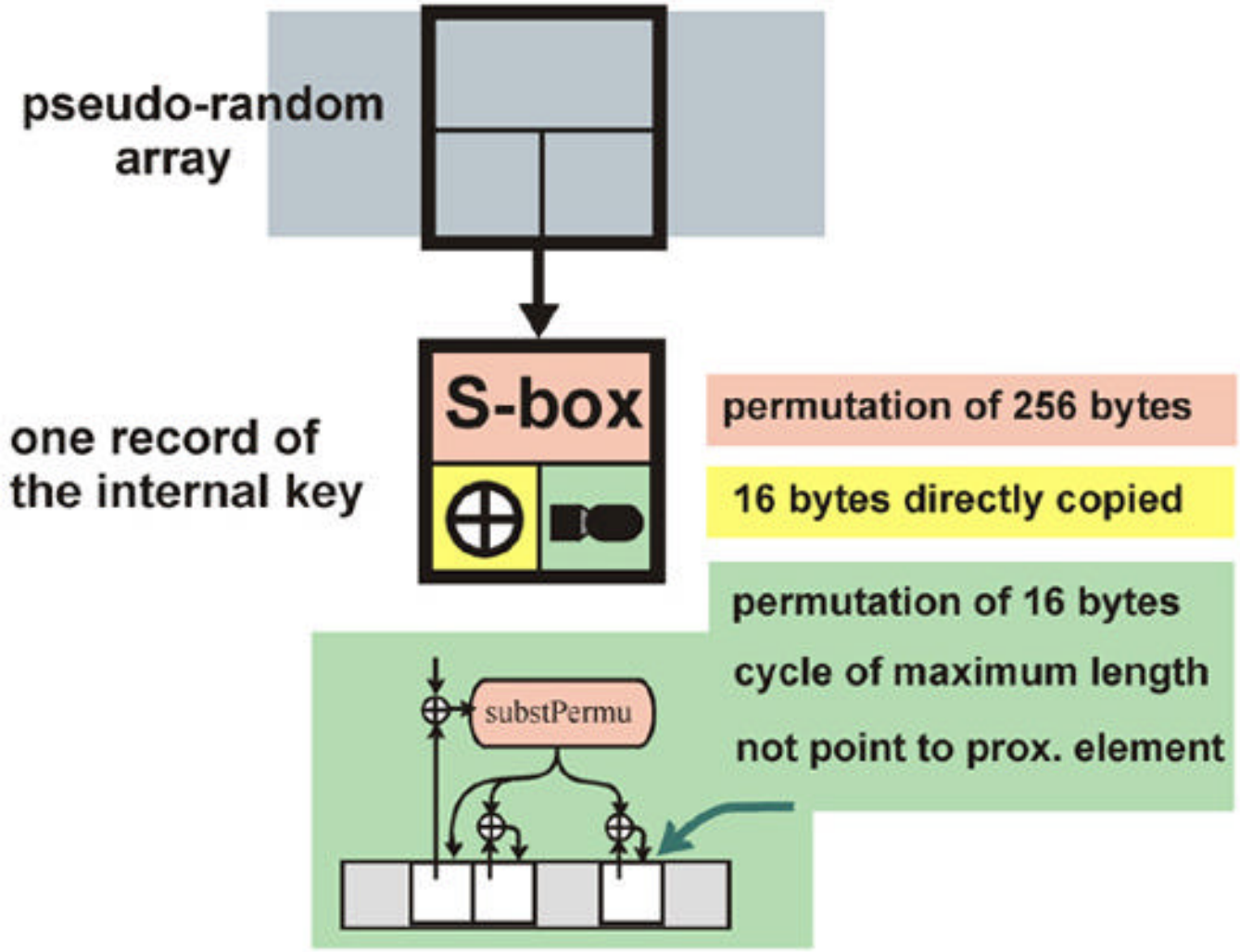
High Level of View of FROG



One Iteration of FROG



Formatting the Internal Key



Permutation Generating Algorithm

Function Transform an array filled with arbitrary bytes into a permutation of the same length.

Algorithm Fill Use-array with sequential values.
Then compute indexes on Use-array:

```
index(i) = ( index(i-1) + P[i]) mod (length-Use-array)
P[i]     = Use[ index(i) ]
remove element index(i) from Use array
```

Example

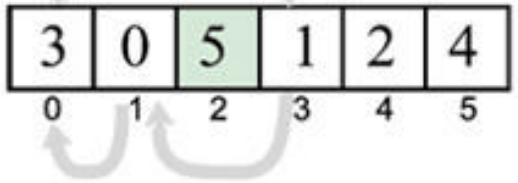
Maximum Cycle Algorithm

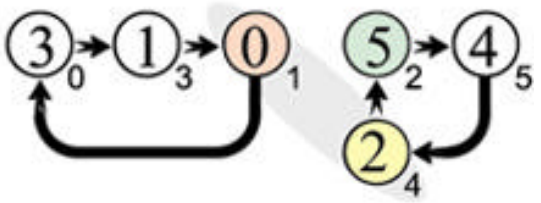
Function Transform an arbitrary permutation into one that has a maximum cycle.

Algorithm If a smaller cycle is found, join it with another cycle. Repeat until only one cycle is present.

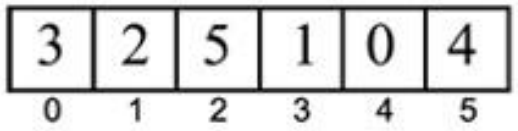
Example

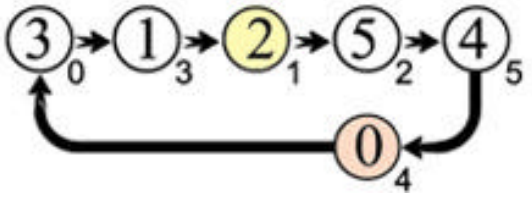
Original state



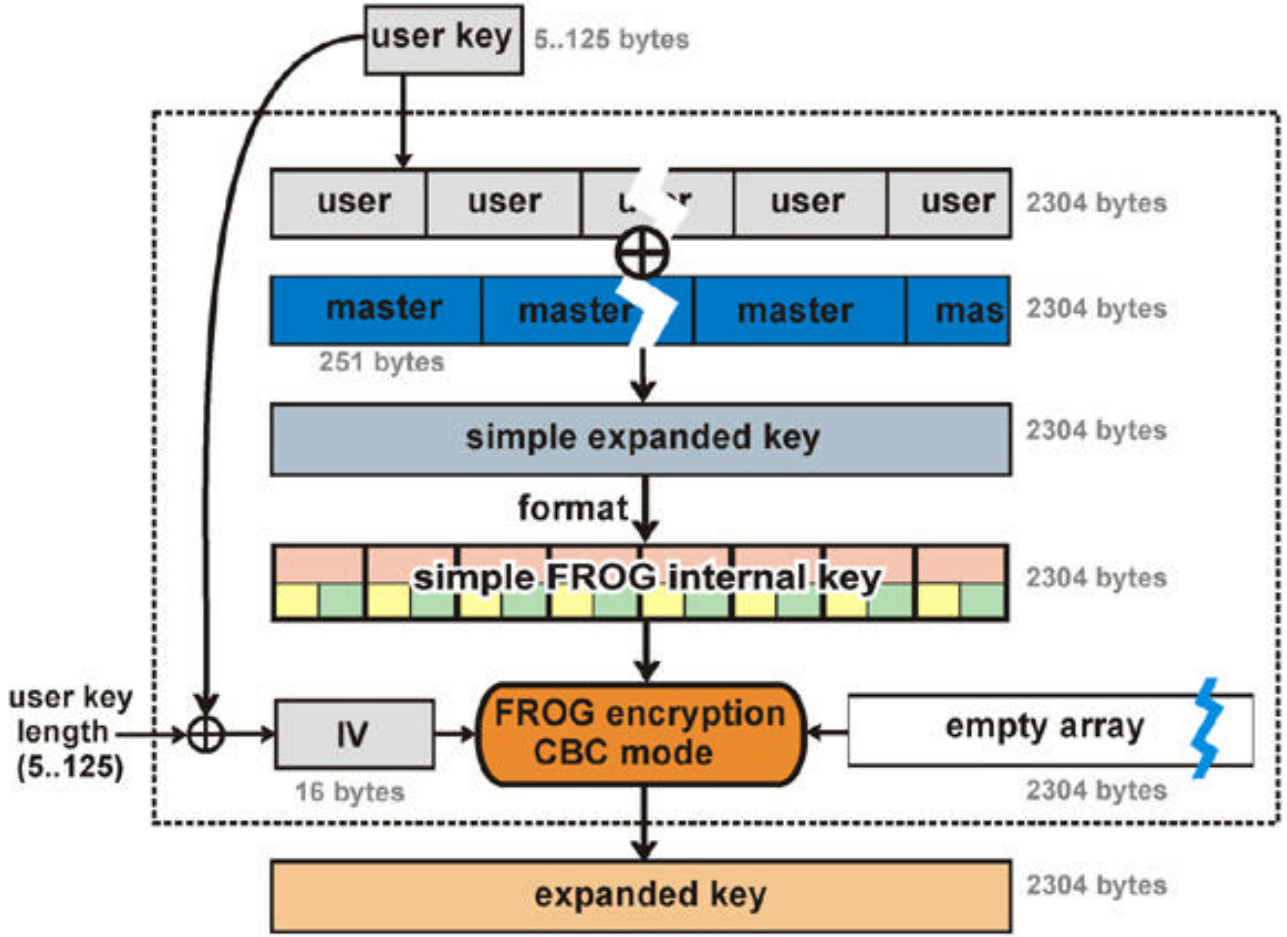


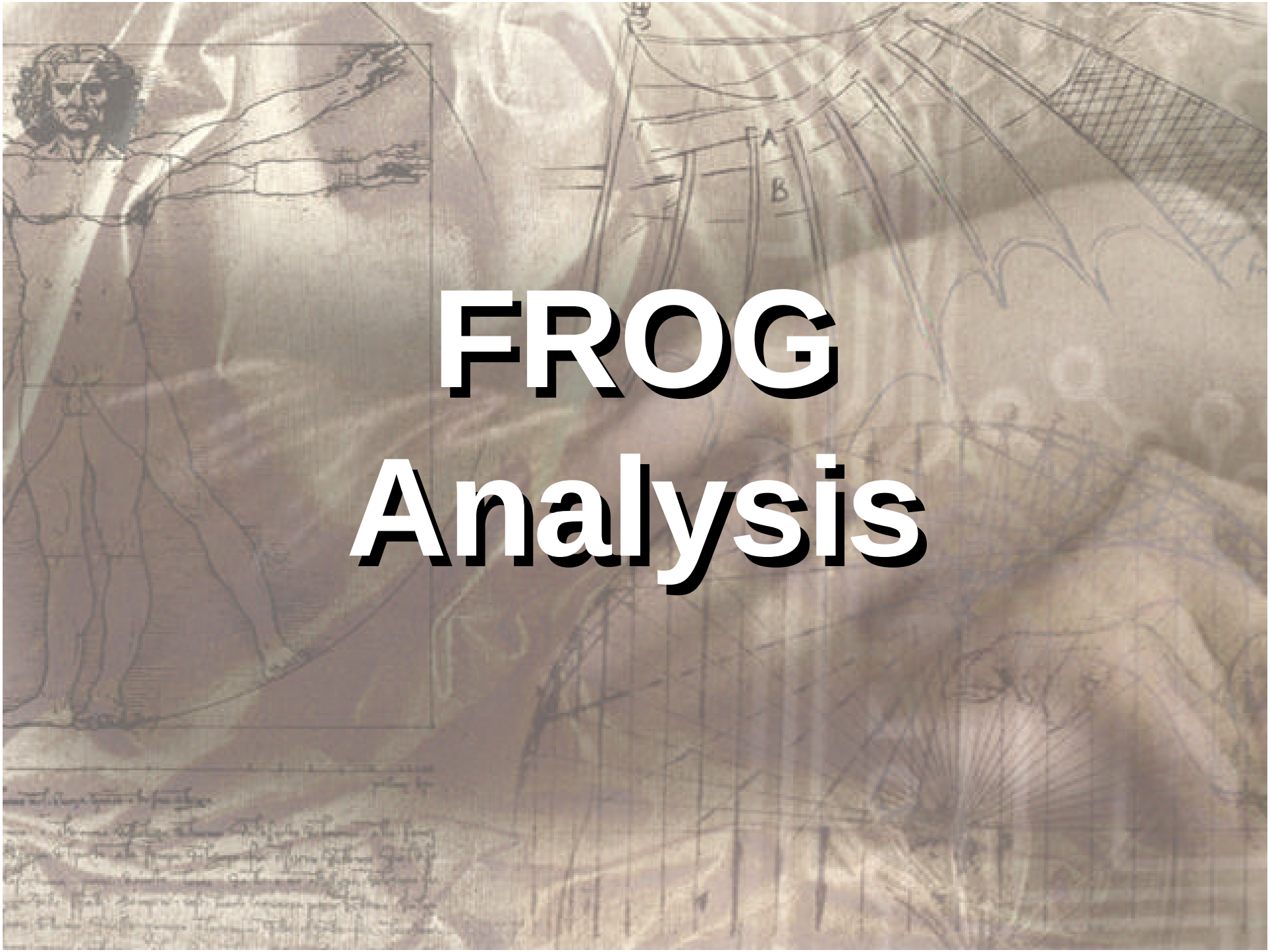
Final state





User Key Hash




FROG
Analysis

Important Characteristics of the Cipher

1. FROG has an extremely simple structure and represents an instance of a novel design principle.
2. FROG uses only byte level XORs and byte level substitutions.
3. FROG's can encrypt blocks of any size between 8 and 128 bytes.
4. FROG's user key can have any size between 5 and 125 bytes.
5. FROG's key expansion can be modified by the user without affecting its strength.
6. FROG has only one constant table whose values are not critical for its strength.
7. No weak keys are known for FROG.
8. Once the internal key is setup, the encryption and decryption processes of FROG are extremely simple (22 machine instructions of 8086 assembler).

TecApro

FROG's strength

FROG is a new type of cipher, designed to resist both known and unknown attacks.

Known attacks such as differential and linear cryptanalysis attacks are not expected to work because FROG uses S-boxes initialized with effectively random values which are unknown to the cryptanalyst.

FROG resists unknown attacks because:

1. It has a simple structure that decreases the probability of a flaw.
2. FROG has not been designed specifically to resist known attacks. If it manages to resist the intensive cryptanalysis that the AES candidates will go through, confidence will be gained about its strength against all attacks.
3. It resists mathematical modeling.
4. It has a very complex internal key setup.

TecApro

FROG's speed

		Pentium 200 MHz		
	8 bit processors	optimized C	8086 assembly	multi-encryption
Machine instructions per byte	~ 100	~ 100	60	150 ?
bytes per second	2.1 K ?	1.3-1.7 M	2.2 M	10 M ?
CPU cycles per byte	1100 ?	120-160	90	20 ?

FROG's very complex key setup process needs 10 mseg on a Pentium 200.

Smart card applications will pre-compute and store the internal key.

FROG's encryption or decryption speed per byte does not depend either on the user key size or on the size of the data block.

TecApro

FROG's memory requirements

FROG memory requirement for encryption or decryption is exactly the size of its internal key:

- **2,304 bytes for 16 byte blocks (NIST standard)**
- **4,096 bytes for 128 byte blocks (largest block tested)**

The key setup process is recursive and requires approximately twice as much memory.

TecApro


FROG
Methodology

FROG's Design Principles

1. Hide as much information about the actual computational process of encryption as possible.
2. Implement an algorithm that is very simple so that no trap-doors can be hidden in it and so that the probability of obscure structural flaws is minimized. (Minimize human bias in the design of the algorithm.)
3. Devise an algorithm that resists mathematical modeling.

The goal of these design principles is to strengthen FROG against all possible attacks, not just known ones.

Generalization of FROG

A more general version of FROG can be created that divulges even less structural information.

1. **FROG processes each byte in the data block sequentially. This can be variable and change in each round.**
2. **FROG uses one S-table per round. Make this variable per byte and per round.**
3. **FROG uses 8 S-tables and 8 rounds. Allow for a variable number of tables and a variable number of rounds.**
4. **FROG's implements key dependent destinations. Allow for data dependent destinations too.**

These points can be easily coded in the current version of FROG without much effect on its speed.

TecApro

Towards a Perfect Vacuum

Even more information can be hidden:

1. **FROG uses only XOR arithmetic operations. Allow for a bigger set that includes ADDs, data dependent circular shifts, etc. Allow the user to define the subset desired.**

2. **FROG processes each byte executing four specific instructions in a fixed order. Allow for a variable number, type and order of instructions. Allow for different "micro-ciphers" for different cycles.**

These points can be implemented efficiently by using a "user-key compiler" that inputs a key and produces executable code and tables.

Generalized FROG works as a cipher generator and is very malleable. For example, if memory is a limited quantity, fewer S-boxes can be used. At the extreme FROG could produce a key-dependent cipher that requires no memory at all.

TecApro


A
B
C
dianelos@tecapro.com